


[> home](#) [> about](#) [> feedback](#) [> login](#)

US Patent & Trademark Office



Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: [patch<AND>((allocation<AND>((scratch register))))]

Found 9 of 122,228 searched.

Search within Results

 [> Advanced Search](#)
[> Search Help/Tips](#)

Sort by: [Title](#) [Publication](#) [Publication Date](#) [Score](#) [Binder](#)

Results 1 - 9 of 9 [short listing](#)

1 [Fast, effective code generation in a just-in-time Java compiler](#) 80%

Ali-Reza Adl-Tabatabai , Micha? Cierniak , Guei-Yuan Lueh , Vishesh M. Parikh , James M. Stichnoth

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation May 1998
Volume 33 Issue 5

A "Just-In-Time" (JIT) Java compiler produces native code from Java byte code instructions during program execution. As such, compilation speed is more important in a Java JIT compiler than in a traditional compiler, requiring optimization algorithms to be lightweight and effective. We present the structure of a Java JIT compiler for the Intel Architecture, describe the lightweight implementation of JIT compiler optimizations (e.g., common subexpression elimination, register allocation, and elim ...

2 [Performance monitoring: METRIC: tracking down inefficiencies in the memory hierarchy via binary rewriting](#) 77%

Jaydeep Marathe , Frank Mueller , Tushar Mohan , Bronis R. de Supinski , Sally A. McKee , Andy Yoo

In this paper, we present METRIC, an environment for determining memory inefficiencies by examining data traces. METRIC is designed to alter the performance behavior of applications that are mostly constrained by their latency to resolve memory references. We make four primary contributions in this paper. First, we present methods to extract partial data traces from running applications by observing their memory behavior via dynamic binary rewriting. Second, we present a methodology to represent ...

3 [Recompilation for debugging support in a JIT-compiler](#) 77%

Mustafa M. Tikir , Jeffrey K. Hollingsworth , Guei-Yuan Lueh


ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and

engineering November 2002**Volume 28 Issue 1**

A static Java compiler converts Java source code into a verifiably secure and compact architecture-neutral intermediate format, called Java *byte codes*. The Java byte codes can be either interpreted by a Java Virtual Machine or translated into native code by Java Just-In-Time compilers. Static Java compilers embed debug information in the Java class files to be used by the source level debuggers. However, the debug information is generated for architecture independent byte codes and most o ...

4 Dynamo: a transparent dynamic optimization system

77%

 Vasanth Bala , Evelyn Duesterwald , Sanjeev Banerjia


ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation May 2000

Volume 35 Issue 5

We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the ...

5 Annotation-directed run-time specialization in C

77%

 Brian Grant , Markus Mock , Matthai Philipose , Craig Chambers , Susan J. Eggers


ACM SIGPLAN Notices , Proceedings of the 1997 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation December 1997

Volume 32 Issue 12

We present the design of a dynamic compilation system for C. Directed by a few declarative user annotations specifying where and on what dynamic compilation is to take place, a binding time analysis computes the set of run-time constants at each program point in each annotated procedure's control flow graph; the analysis supports program-point-specific polyvariant division and specialization. The analysis results guide the construction of a specialized run-time specializer for each dynamically c ...

6 VCODE: a retargetable, extensible, very fast dynamic code generation system

77%

 Dawson R. Engler

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation May 1996

Volume 31 Issue 5

Dynamic code generation is the creation of executable code at runtime. Such "on-the-fly" code generation is a powerful technique, enabling applications to use runtime information to improve performance by up to an order of magnitude [4, 8, 20, 22, 23]. Unfortunately, previous general-purpose dynamic code generation systems have been either inefficient or non-portable. We present VCODE, a retargetable, extensible, very fast dynamic code generation system. An important feature of VCODE is that it ge ...

7 Exokernel: an operating system architecture for application-level resource management


77%

 D. R. Engler , M. F. Kaashoek , J. O'Toole

ACM SIGOPS Operating Systems Review , Proceedings of the fifteenth ACM

symposium on Operating systems principles December 1995
Volume 29 Issue 5

8 Reducing virtual call overheads in a Java VM just-in-time compiler 77%


 Junpyo Lee , Byung-Sun Yang , Suhyun Kim , Kemal Ebcioglu , Erik Altman , Seungil Lee , Yoo C. Chung , Heungbok Lee , Je Hyung Lee , Soo-Mook Moon

ACM SIGARCH Computer Architecture News March 2000

Volume 28 Issue 1

Java, an object-oriented language, uses *virtual methods* to support the extension and reuse of classes. Unfortunately, virtual method calls affect performance and thus require an efficient implementation, especially when just-in-time (JIT) compilation is done. *Inline caches* and *type feedback* are solutions used by compilers for dynamically-typed object-oriented languages such as SELF [1, 2, 3], where virtual call overheads are much more critical to performance than in Java. We ...

9 Poor man's watchpoints 77%

 Max Copperman , Jeff Thomas

ACM SIGPLAN Notices January 1995

Volume 30 Issue 1

Bugs that result from corruption of program data can be very difficult to track down without specialized help from a debugger. If the debugger cannot help the user find the point at which data gets corrupted, the user may have a long iterative debugging task. If the debugger is able to stop execution of the program at the point where data gets corrupted, as with watchpoints (also known as data breakpoints), it may be a very simple task to find a data corruption bug. In this paper, we discuss a m ...

Results 1 - 9 of 9 [short listing](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2003 ACM, Inc.

**IEEE Xplore**
RELEASE 1.5Welcome
United States Patent and Trademark Office[Help](#) [FAQ](#) [Terms](#) [IEEE](#) [Quick Links](#)[» Search Results](#)[Peer Review](#)

Welcome to IEEE Xplore™

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

Print Format

Your search matched **2** of **981130** documents.A maximum of **2** results are displayed, **15** to a page, sorted by **Relevance** in **descending** order.

You may refine your search by editing the current search expression or entering a new one in the text box.

Then click **Search Again**.

(register allocation)and (indexing)

[Search Again](#)**Results:**Journal or Magazine = **JNL** Conference = **CNF** Standard = **STD****1 Handling cross interferences by cyclic cache line coloring***Genius, D.;*

Parallel Architectures and Compilation Techniques, 1998. Proceedings.

1998 International Conference on , 12-18 Oct. 1998

Page(s): 112 -117

[\[Abstract\]](#) [\[PDF Full-Text \(232 KB\)\]](#) **IEEE CNF****2 Optimized array index computation in DSP programs***Leupers, R.; Basu, A.; Marwedel, P.;*

Design Automation Conference 1998. Proceedings of the ASP-DAC '98.

Asia and South Pacific , 10-13 Feb. 1998

Page(s): 87 -92

[\[Abstract\]](#) [\[PDF Full-Text \(596 KB\)\]](#) **IEEE CNF**

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#)
[Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#)
[No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

Copyright © 2003 IEEE — All rights reserved

Welcome to IEEE Xplore

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

 Print FormatYour search matched **4** of **981130** documents.A maximum of **4** results are displayed, **25** to a page, sorted by **Relevance** in **descending** order.

You may refine your search by editing the current search expression or entering a new one the text box.

Then click **Search Again**.

(register)and (instrumentation) and (allocation)

Results:

Journal or Magazine = **JNL** Conference = **CNF** Standard = **STD****1 Code generation for a DSP processor***Wei-Kai Cheng; Youn-Long Lin;*

High-Level Synthesis, 1994., Proceedings of the Seventh International Symposium on , 18-20 May 1994

Page(s): 82 -87

[\[Abstract\]](#) [\[PDF Full-Text \(468 KB\)\]](#) **IEEE CNF****2 Automatic generation of optimized DSP assembly code***Wess, B.; Kreuzer, W.; Gotschlich, M.;*

Industrial Electronics, Control, and Instrumentation, 1995., Proceedings of the 1995 IEEE IECON 21st International Conference on , Volume: 2 , 6-10 Nov. 1995

Page(s): 979 -984 vol.2

[\[Abstract\]](#) [\[PDF Full-Text \(468 KB\)\]](#) **IEEE CNF****3 Considerations for implementing high performance VXI test systems***Emmert, G.T.;*

AUTOTESTCON '98. IEEE Systems Readiness Technology Conference., 1998 IEEE , 24-27 Aug. 1998

Page(s): 466 -473

[\[Abstract\]](#) [\[PDF Full-Text \(904 KB\)\]](#) **IEEE CNF****4 Discrete wavelet transform architecture using fast**

processing elements

processing elements*Huluta, E.; Petriu, E.M.; Das, S.R.; Al-Dhafer, A.H.;*

Instrumentation and Measurement Technology Conference, 2002.

IMTC/2002. Proceedings of the 19th IEEE , Volume: 2 , 21-23 May 2002

Page(s): 1537 -1542 vol.2

[\[Abstract\]](#) [\[PDF Full-Text \(518 KB\)\]](#) **IEEE CNF**

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#)
[Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#)
[No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

Copyright © 2003 IEEE — All rights reserved